LAB - 8

1. Implement Johnson Trotter algorithm to generate permutations.

#include <stdio.h>

#include <stdlib.h>

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void generatePermutations(int arr[], int start, int end) {

if (start == end) {

for (int i = 0; i <= end; i++) {

printf("%d ", arr[i]);

}

printf("\n");

} else {

for (int i = start; i <= end; i++) {

swap(&arr[start], &arr[i]);

generatePermutations(arr, start + 1, end);

swap(&arr[start], &arr[i]); }

}

}

int main() {

int n;

printf("Enter the number of elements: ");

scanf("%d", &n);

int\* arr = (int\*)malloc(n \* sizeof(int));

printf("Enter the elements: ");

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

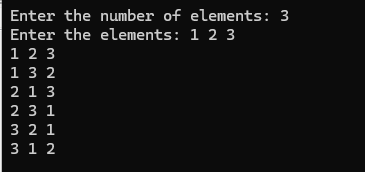
generatePermutations(arr, 0, n - 1);

free(arr);

return 0;

}

**OUTPUT-**

****

2. Implement 0/1 Knapsack problem using dynamic programming.

#include <stdio.h>

int n,m,w[10],p[10],v[10][10];

void knapsack(int,int,int[],int[]);

int max(int,int);

int main()

{

int i,j;

printf("Enter the no. of items:");

scanf("%d",&n);

printf("Enter the capacity of knapsack:");

scanf("%d",&m);

printf("Enter weights:");

for(i=0;i<n;i++){

scanf("%d",&w[i]);

}

printf("Enter profits:");

for(i=0;i<n;i++){

scanf("%d",&p[i]);

}

knapsack(n,m,w,p);

printf("Optimal Solution:\n");

for(i=0;i<n;i++){

for(j=0;j<n;j++){

printf("%d ",v[i][j]);

}

printf("\n");

}

return 0;

}

void knapsack(int n, int m, int w[],int p[]){

int i,j;

for(i=0;i<n;i++){

for(j=0;j<m;j++){

if(i==0 || j==0){

v[i][j]=0;

}else if(w[i]>j){

v[i][j]=v[i-1][j];

}else{

v[i][j]=max(v[i-1][j],((v[i-1][j-w[i]])+p[i]));

}

}

}

}

int max(int a,int b){

if(a>b){

return a;

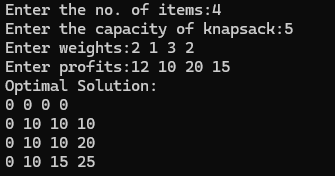
}else{

return b;

}

}

**OUTPUT-**

****